INTRODUCTION TO ADO.NET

DEPARTMENT OF INFORMATION TECHNOLOGY

Compiled by: Afaq Alam Khan

ADO.Net

- Introduction
- Data Providers
  - Connection
  - Command
  - DataReader
  - DataAdapter
- Dataset
- Demo

Introduction

- ADO.NET is a large set of .NET classes that enable us to retrieve and manipulate data, and update data sources, in many different ways.
- ADO.NET is the latest in a long line of data access technologies released by Microsoft (ODBC, DAO, RDO, OLE DB, ADO).
- ADO.NET differs somewhat from the previous technologies, however, in that it comes as part of a platform called the .NET Framework.
- Just as .NET includes a library of classes for managing rich client UI (Windows Forms) and for handling HTTP requests (ASP .NET), .NET includes a library for connecting to a wide range of databases .That library is named ADO.NET.

- ADO.NET has a couple of new ways to serve data, which made the Recordset (available in ASP) obsolete. These new objects are the **DataSet** *(disconnected)* and the **DataReader** *(always connected)*.

.Net Data Providers

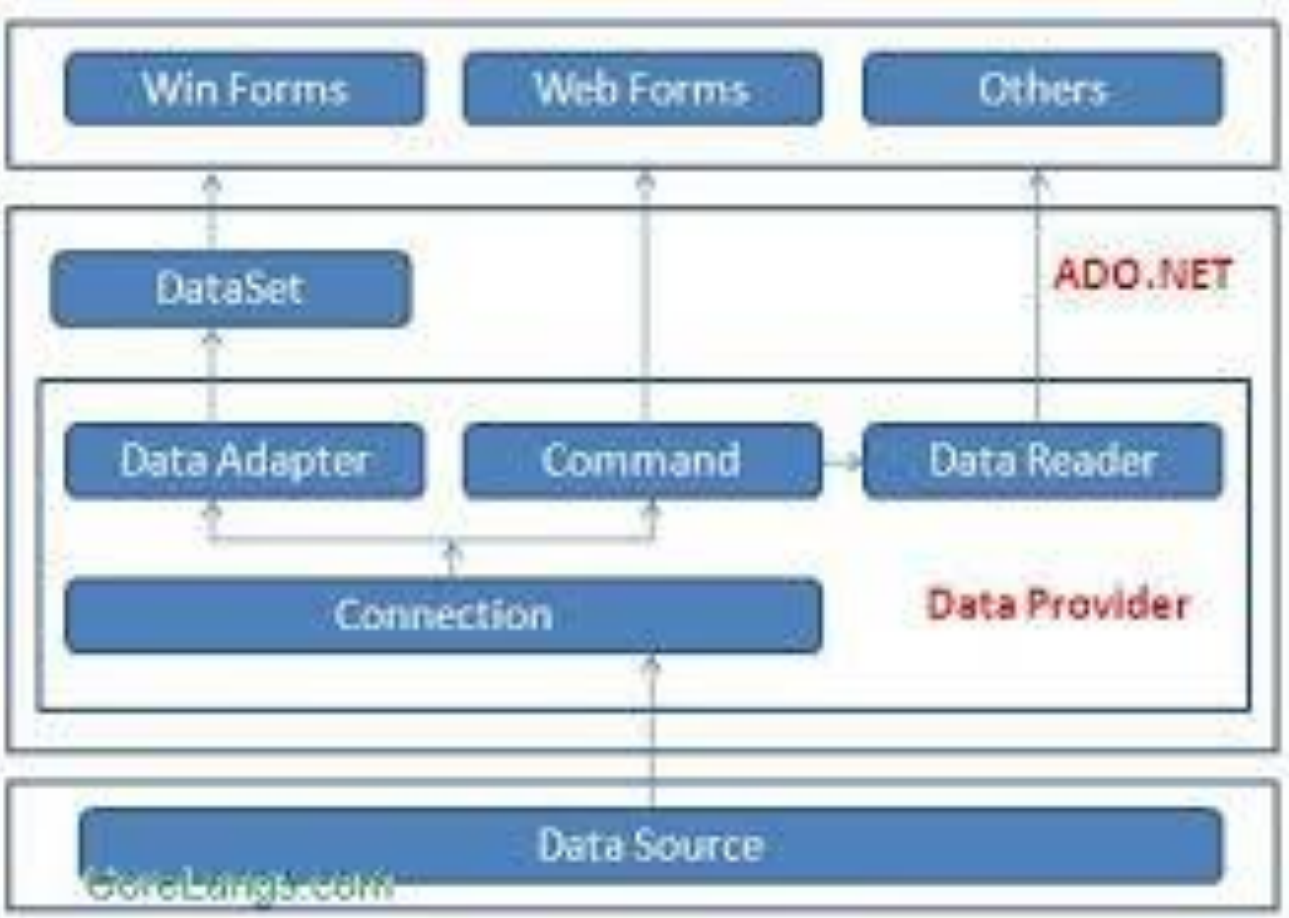
- The .NET data providers allow us to connect to the data source, and to execute SQL commands against it.
- Some .NET data providers available:
  - for SQL Server,
  - for OLE DB data sources {Object Linking & Embedding}
  - for ODBC-compliant data sources. {Object Database connectivity}
  - for Oracle data source

- Each provider exists in a namespace within the System.Data namespace, and consists of a number of classes.
  - E.g., System.Data.SqlClient
  - System.Data.Oledb
  - System.Data.OracleClient

Data Provider Components

- Each .NET data provider consists of four main components:
  - **Connection** – used to connect to the data source
  - **Command** – used to execute a command against the data source and retrieve a DataReader or DataSet, or to execute an INSERT, UPDATE, or DELETE command against the data source
  - **DataReader** – a forward-only, read-only connected resultset
  - **DataAdapter** – used to populate a **DataSet** with data from the data source, and to update the data source

Architecture



Connection

- The connection classes store the information that ADO.NET needs to connect to a data source in the form of connection string.
- ConnectionString property holds information such as the username and password of the user, the name and location of the data source to connect to, and so on.
- The connection classes also have methods for opening and closing connections, and for beginning a transaction, and properties for setting the timeout period of the connection and for returning the current state (open or closed) of the connection.
- connection should be opened as late as possible and closed as soon as possible.
- ADO.Net provides **SQLConnection** class for sqlserver data source and **oledbconnection** class for other data sources

Connection String

- The connection string is a list of key/value pairs that the Connection object will parse; it will use the information to find the Data Source, authenticate, and establish a connection.
- Depending on the namespace used, the connection string will vary a little. Basically the connection string for a SqlConnection does not have the Provider attribute, while the connection string for an OleDbConnection does.

Data Source/ server	The name or IP address of the SQL Server to make the connection with
Initial Catalog / Database	The name of the database. If this is not specified you will get a connection to the default database defined for the User ID.
Integrated Security/ Trusted Connection	Whether SQL Server will use the NT user credentials, or expect a SQL Server username and password. (default 'false') /set to SSPI (Security Support Provider Interface) for windows authentication
User ID	The SQL Server login account.
Password	The password for the SQL Server account logging on. For integrated security, this is not specified.
Persist Security Info	When set to 'false,' security sensitive information, such as the password, is not returned as part of the connection if the connection is open or has ever been in an open state. Resetting the connection string resets all connection string values including the password.

Example

- Connection string for MSAccess Datasource
  - Using oledbConnection
    - Provider=Microsoft.ACE.OLEDB.12.0;Data Source="C:\Users\cuk\Documents\Visual Studio 2013\Projects\ditweb1\cukitdDB.accdb“

- Connection string for sqlserver
  - Using sqlconnection
    - “server=localhost;database=Northwind;uid=sa;pwd=;“
  - Using oledbconnection
    - "Provider=SQLOLEDB; Data Source=localhost; Initial Catalog=Northwind; Integrated Security=SSPI;“
  - http://www.dofactory.com/reference/connection-strings

- Methods of sqlConnection/oledbconnection class
  - Open()
  - Close()

Storing Connection string in configuration file

- **The connection string can be put into the web.config file as shown below:**

```
<connectionStrings>
<add name="myConnectionString"
connectionString="server=localhost;database=myDb;uid=myUser;password=
myPass;" />
</connectionStrings>
```

All the connection strings defined in the web.config file are loaded into the new ConfigurationManager.ConnectionStrings collection. To physically open a connection based on a string stored in the web.config file, use following code:

```
string connStr =
ConfigurationManager.ConnectionStrings("myConnectionString“).Connec
tionString;
```

Command

- The Command is used to store SQL statements that need to be executed against a data source.
- The Command object can execute SELECT statements, INSERT, UPDATE, or DELETE statements, stored procedures, or any other statement understood by the database.
- **SqlCommand** for sqlserver data sources and **oledbCommand** for other datasources
- Properties
  - CommandText: sql command
  - Connection: connection object
- The Command object has three basic methods:
  - **ExecuteReader:** Used to execute an SQL SELECT query and get the result through a DataReader object.
  - **ExecuteNonQuery**: Used to execute any SQL query and it returns the number of rows affected in the database.
  - **ExecuteScaler**: Used to execute any SQL query which usually returns a single value, e.g. the aggregate functions.

- The command classes (sqlcommand/oledbCommand) expose a CommandType property. By default, this property is set to "CommandType.Text", indicating that a T-SQL statement will be used as the CommandText property.
- Possible values for the CommandType property are:
  - CommandType.Text – a SQL statement
  - CommandType.TableDirect – a table name whose columns are returned
  - CommandType.StoredProcedure – the name of a stored procedure

Example (delete query) [C#.Net]

```
Using System
Using System.Data
Using System.Data.sqlClient;
.
.
SqlConnection con = new
    SqlConnection("server=localhost;database=Northwind;uid=sa;pwd=;");
SqlCommand cmd = new SqlCommand();
 cmd.CommandText = "DELETE FROM Customers WHERE CustomerID = 'SEVEN'";
 cmd.Connection = con;
con.Open();
cmd.ExecuteNonQuery();
Con.Close();
```

Example (Insert Query) [VB.Net]

```
Imports System
Imports System.data
Imports System.data.oledb
.
.
Dim Conn As oledbConnection
Conn = new oledbConnection(“                ”)
Dim Cmd As oledbCommand
Cmd = new oledbCommand(“ Insert .....      ”, conn)
Conn.open()
Cmd.ExecuteNonQuery()
Conn.Close()
```

Example [Use of Parameters] (Insert)

```
con.Open();
cmd = new SqlCommand("insert into Comment (UserName,Subject,CommentOn,Post_Date)
    values(@userName,@subject,@comment,@date)", con);
cmd.Parameters.Add("@userName", SqlDbType.NVarChar).Value = txtName.Text.ToString();
cmd.Parameters.Add("@subject", SqlDbType.NVarChar).Value = txtSubject.Text.ToString();
 cmd.Parameters.Add("@comment", SqlDbType.NVarChar).Value =
    txtComment.Text.ToString();
cmd.Parameters.Add("@date", SqlDbType.DateTime).Value = DateTime.Now.Date;
 cmd.ExecuteNonQuery();
   con.Close();
```

DataReader

- The datareader is a forward-only, readonly stream of data from the database
- data reader are manifested as either a SqlDataReader or an OleDbDataReader
- datareader is a very efficient means for retrieving data, as only one record is brought into memory at a time
- Calling Command.ExecuteReader() will return a datareader object. We can use the DataReader.Read() method to iterate through the rows in the returned stream of data. The Read() method will advance to the next record in the stream.
- Each datareader (OleDbDataReader and SqlDataReader) exposes several methods for retrieving the values of the record fields, each of which takes the zero-based column ordinal as its input argument
- A connection object can only contain one datareader at a time, so we must explicitly close the datareader when we are done with it. This will free the connection for other uses.

- Each datareader object exposes a different set of methods, but the following list are methods exposed by both the OleDbDataReader and the SqlDataReader:
- Properties
  - FieldCount
  - IsClosed
  - Item
  - HasRows
  - .
  - .

Methods of DataReader

- **Close()**
- **Read():** Reads next record in datareader
- **NextResult():** Advances the data reader to the next result during the batch transaction
- **GetValue():** Gets the value of specified column in its native format
- **GetValues():** Populates an array of objects with the column values of current row
- **GetOrdinal():** Gets column ordinal, given the name of column
- **GetName():** Gets the name of specified column
- .
- .

Example

```
While reader.Read()
   txtResults.Text = txtResults.Text & reader("fname") & _
      ControlChars.Tab & reader("lname") & _
      ControlChars.Tab & ControlChars.Tab & _
      reader("job_desc") & ControlChars.CrLf
End While
reader.Close()
```

Example

This program takes two numbers RowNo and ColNo from the user through textboxes and displays in label control the data item available in datareader at the asked location.

```
rowNo = TextBox1.Text
 ColNo = TextBox2.Text
currentRow=0


  conn.Open()
   Dr = cmd.ExecuteReader()
  While Dr.Read()
   If (CurrentRow = rowNo) Then
   Label1.Text = Dr.GetValue(ColNo).ToString
   End If
   CurrentRow = CurrentRow + 1
   End While
   Dr.Close()
   conn.Close()
```

Example

Adding data items from datareader to ListBox Control

.

.

```
While reader.Read()
ListBox1.Items.Add(reader.Item(0) & " - " & reader.Item(1))
End While
 reader.Close()
```

.

.

Example (Bind data to dropdownlist control) using oledbDataReader

```
Imports system
Imports system.data
Imports syste.data.oledb
.
.
Dim conn  As oledbConnection
Dim cmd As oledbCommand
Dim Dr As oledbDataReader
Conn = new oledbConnection(“        ”)
Cmd = new oledbCommand(“ Select * from country”, Conn)
Conn.open()
Dr = Cmd.ExecuteReader()

DropDownList1.Datasource =Dr
DropDownList1.DataTextField = “CountryName”
DropDownList1.DataValueField=“CountryID”
DropDownList1.Databind()
Dr.close()
Conn.close()
```

Example (Binding Gridview Control) using oledbDataReader

```
Imports system
Imports system.data
Imports syste.data.oledb
.
.
Dim conn  As oledbConnection
Dim cmd As oledbCommand
Dim Dr As oledbDataReader
Conn = new oledbConnection(“        ”)
Cmd = new oledbCommand(“ Select * from  studentstbl”, Conn)
Conn.open()
Dr = Cmd.ExecuteReader()

GridView1.Datasource =Dr
GridView1.Databind()
Dr.close()
Conn.close()
```

Data Adapter

- DataAdapter provides a bridge between a DataSet and the database.
- The DataAdapter transfers data from a data source to a dataset, and back again – a key ability of ADO.NET.
- The DataAdapter uses command objects to execute SELECT, INSERT, UPDATE, and DELETE commands against the database and return any requested data into, or reconcile data from, a DataSet.
- The DataAdapter enables the DataSet to update a database using SQL commands stored within the InsertCommand, UpdateCommand, and DeleteCommand properties.
- Data Adapter is manifested as **SqlDataAdapter** and **OleDbDataAdapter**
- The DataAdapter is intended for use with a DataSet and can retrieve data from the data source, populate DataTables and constraints, and maintain the DataTable relationships.
- The DataAdapter exposes a Fill() method, which is used to fill a DataTable in a DataSet. The Fill method takes in a DataSet instance as a required argument and a table name as an optional argument. If a table name is not specified, the DataTable will be given a default name

Example (bind Gridview to datasource using dataset/dataAdapter [C#]

```
using System;
using System.Data;
 using System.Data.SqlClient;
.
.
 DataSet ds = new DataSet();
 SqlDataAdapter sda = new SqlDataAdapter ( "SELECT * FROM Customers",
     "server=localhost;database=Northwind;uid=sa;pwd=;");
 sda.Fill(ds, "Customers");

GridView1.Datasource =ds;
GridView1.DataBind();
.
.
```

When we invoke the sda.Fill() method, the connection is opened, the command is executed, a DataTable named "Customers" is created in the DataSet, and the connection is closed.

DataSet

- DataSet is a disconnected representation of the database. A DataSet contains DataTables, Relations, and Constraints, allowing it to replicate the entire database, or select parts of the database, in a disconnected fashion, using XML as its underlying persistence format.
- The data in the DataSet can be manipulated – changed, deleted, or added to – without an active connection to the database.

Example – Bind DropDownList control to database using dataset [vb.net]

```
Dim DA As OleDbDataAdapter
    Dim DS As DataSet
    Dim constr As String
    constr = "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\cuk\Documents\Visual Studio
2008\WebSite1\cukitdDB.accdb"
    DA = New OleDbDataAdapter("select namee from stdinfo", constr)
    DS = New DataSet()
    DA.Fill(DS, "stdinfo")
    DropDownList1.DataSource = DS
    DropDownList1.DataTextField = "namee"
    DropDownList1.DataBind()
```

Updating changes in Dataset

- DA.Update(DS, “Customers”)
- Before Submitting the update back to database, we need to setup the InsertCommand, UpdateCommand and DeleteCommand to reconcile the changes to the Database.
- We can also use SqlCommandBuilder/ oledbCommandBuilder to automatically generate those commands.

Updating Record in a **Dataset** and applying modifications back to **database**

```
Dim CB As OleDbCommandBuilder
CB = New OleDbCommandBuilder(DA)
DS.Tables("stdinfo").Rows(inc).Item(1) = TextBox1.Text
DS.Tables("stdinfo").Rows(inc).Item(2) = TextBox2.Text
DA.Update(DS, "stdinfo")
Label1.Text = "Date items updated“
```

We pass our DataAdapter into the OleDbCommandBuilder's constructor, so it is automatically connected to our data source.

By instantiating the CommandBuilder for this DataAdapter – this will automatically build the commands for us. We can retrieve the text of the generated commands by calling the CommandBuilder's GetUpdateCommand, GetInsertCommand, etc., methods

Inserting Record in a **Dataset** and applying modifications back to **database**

```
Dim CB As oledbCommandBuilder
CB = New oledbCommandBuilder(DA)
Dim DSNewRow As DataRow
DSNewRow = DS.Tables("Stdinfo").NewRow()
DSNewRow.item("FirstName") = TxtFirstName.Text
DSNewRow.item("Cellphone") = TxtCellphone.Text
DS.Tables("Stdinfo").Rows.Add(DSNewRow)
DA.UpDate(DS,"Stdinfo")
```

```
DSNewRow.item(0) = TxtFirstName.Text
DSNewRow.item(1) = TxtCellphone.Text
```

Deleting Record in a **Dataset** and applying modifications back to **database**

```
Dim CB As OleDbCommandBuilder
 CB = New OleDbCommandBuilder(DA)
 DS.Tables("stdinfo").Rows(inc).Delete()
   DA.Update(DS, "stdinfo")
 Label1.Text = "Row Deleted Successully"
```

Navigating through rows of dataset

```
Inc= 0

Maxrows = DS.Tables("stdinfo").Rows.Count
‘ Accessing Next Row in the dataset
If (inc <> Maxrows - 1) Then
 inc = inc + 1
 TextBox1.Text = DS.Tables("stdinfo").Rows(inc).Item(1)
 TextBox2.Text = DS.Tables("stdinfo").Rows(inc).Item(2)
  Else
 Label1.Text = "All rows done“
  End If
```

Navigating through rows of dataset

```
.
.
‘ accessing previous row in the dataset

If (inc >= 1) Then
 inc = inc – 1
TextBox1.Text = DS.Tables("stdinfo").Rows(inc).Item(1)
 TextBox2.Text = DS.Tables("stdinfo").Rows(inc).Item(2)
 Else
Label1.Text = "This is the first row“
 End If
```

Self study

- Working with stored procedures
- Repeater Control

References

- B. Joshi, P. Dickinson, ... Professional ADO.Net Programming, Wrox Publ,